Санкт-Петербургский Государственный Университет

Математическое Обеспечение и Администрирование Информационных Систем

Кафедра Информационно-Аналитических Систем

Сандул Михаил Вадимович

Автоматическое аннотирование одиночных документов

Бакалаврская работа

Научный руководитель:
доцент, к.ф.-м.н. Михайлова Е.Г.

Рецензент:
профессор, доктор педагогических наук Поздняков С.Н.

Санкт-Петербург
2017

Saint-Petersburg State University

Software and Administration of Information Systems

Sub-Department of Analytical Information Systems

Sandul Mikhail

Automatic Annotation of Single Documents

Bachelor's Thesis

Scientific Supervisor:
Assoc. Prof., Ph.D. Elena Mikhailova

Reviewer:
proffesor, Sc.D. Sergei Pozdnyakov

Saint-Petersburg
2017

Содержание

1 Введение

Анализ структурированной информации, хранящейся в базах данных, требует предварительной обработки: проектирование БД, ввод информации по определенным словам, размещение её в определенных структурах (например, реляционных таблицах) и т.п. Таким образом, непосредственно для анализа этой информации и получения из неё новых знаний необходимо затратить дополнительные усилия. При этом они не всегда связаны с анализом, и не всегда приводят к желаемому результату. Из-за этого КПД анализа структурированной информации снижается. Кроме того, не все виды данных можно структурировать без потери полезной информации. Например, тестовые документы практически невозможно преобразовать в табличное представление без потери семантики текста и отношений между сущностями. По этой причине такие документы хранятся в БД без преобразований, как текстовые поля (BLOB-поля). В то же время в тексте скрыто огромное количество информации, но её неструктурированность не позволяет использовать средства Data Mining. Решением этой проблемы занимаются методы анализа неструктурированного текста. В западной литературе такой анализ называют Text Mining.[1]

В настоящее время много прикладных задач, решаемых с помощью анализа текстовых данных. Это и классические задачи Data Mining: классификация, кластеризация, и характерные только для текстовых документов задачи: автоматическое аннотирование, извлечение ключевых понятий и др.[1]

Существует большое количество неструктурированной текстовой информации на разных языках мира. Благодаря развитию всемирной сети это количество постоянно растёт. Анализ таких данных играет важную роль в получении новой информации. Для её извлечения используются ключевые слова. Они позволяют получить представление о содержании документа, не читая его. Извлечение ключевых слов является одной из основных проблем Text Mining.

Ключевые слова могут быть использованы для расширения функциональности систем информационного поиска. Так, например, система Phrasier использует их, для того, чтобы найти документы близкие исходному по содержанию, а сами вхождения слов выступают в роли ссылок между документами, позволяя пользователю получить быстрый доступ к нужному материалу.[8] Система Keyphind – поисковая система для электронных библиотек. Автоматически извлеченные ключевые фразы используются как основной элемент индексации и представления информации. Ключевые фразы отражают диапазон тем, которые охватывают найденные документы. Эта информация позволяет пользователям лучше ориентироваться в результатах поиска.[9].

Несмотря на столь широкий круг возможностей приложения ключевых слов для решения задач анализа, индексирования и поиска, большинство документов не имеют готовых ключевых слов. Большинство существующих подходов сосредоточены на их ручном обозначении. Обычно эта процедура выполняется специалистами соответствующей области. В своей деятельности они полагаются на принципы классификации и систематизации для конкретной предметной области, а также на суждения и мнения автора текста. Исследования же направлены на автоматизацию этого процесса.

Ранние работы использовали статистические данные для извлечения отдельных слов из коллекции документов. У таких подходов есть свои недостатки. В то время как некоторые слова считались ключевыми в рамках целой коллекции, ключевые слова в рамках одного или нескольких документов из этой коллекции уже могли не попадать под нужную категорию. Кроме того, подобные подходы оперировали только с одиночными словами. Чтобы избежать этих недостатков, дальнейшие исследования были посвящены извлечению ключевых слов из одиночных документов.[2]

Определим ключевые слова как последовательности из одного и более слов, которые наиболее полно отражают содержание текста. Результатом работы алгоритма является список из таких последовательностей, причём необязательно, чтобы они образовывали связный текст.

Эффективность алгоритма оценивается в терминах полноты (recall) и точности (precision).

Полнота определяется как отношение числа правильно извлеченных ключевых слов (true positives) к числу всех ключевых слов в тексте (true positives + false negatives) :

$$\frac{true\ positives}{true\ positives + false\ negativees}$$

. Точность – как отношение числа правильно извлеченных ключевых слов (true positives) к числу всех извлеченных ключевых слов (true positives + false positives) :

$$\frac{true\ positives}{true\ positives + false\ positives}$$

. На практике также используется ещё одна величина – F-мера. Она является производной от полноты и точности и определяется как

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot prcision + recall}$$

. Изменяя параметр β, можно увеличить влияние одной из величин: точности или полноты, на значение F-меры. В работе алгоритмы будут оцениваться только по первым двум характеристикам.

Существуют различные подходы к решению задачи автоматического извлечения ключевых слов. Для этих целей применяются методы машинного обучения, как с учителем, так и без. Статистические подходы и подходы, которые используют лингвистические особенности, также могут решать задачу. Методы можно разбить на группы в зависимости от того, определен ли данный алгоритм для текстов, принадлежащих конкретной предметной области (domain dependent / domain independent). В данной работе будут рассмотрены только алгоритмы, независимые от тренировочного набора и какой-либо области.

2 Постановка задачи

Неструктурированных текстовых данных становится всё больше. Извлечение полезной информации из них – важная задача. Многие подходы к решению задачи автоматического аннотирования были придуманы для англоязычных текстов. Тем не менее, анализ русскоязычных документов также представляет интерес. В связи с этим возникает вопрос об эффективности таких алгоритмов для русскоязычных текстов и целесообразности их применения для извлечения ключевых слов. К сожалению, говорить об одинаковых оценках для разных языков нельзя. Так, например, некоторые алгоритмы могут не учитывать существование нескольких форм одного слова, считая такие формы разными словами. Или использовать лингвистические особенности, которые будут несправедливы вне английского языка. Целью данной работы является рассмотрение нескольких алгоритмов для извлечения ключевых слов и детальный разбор некоторых из них.

3 Подходы к решению задачи

3.1 TextRank[3]

Представления данных в виде графа является одним из подходов в поиске значимой информации. В рамках такой модели определяется важность каждой вершины по принципу "голосования"или "рекомендации". Когда из одной вершины существует ребро в другую, то это может считаться как "голос"в пользу такого соседа. Чем больше голосов за какую-нибудь вершину, тем она важнее. Кроме того, важность "голосующей"вершины определяет важность и её "рекомендации". Основываясь на таких принципах можно определять значимость информации.

Опишем общий подход более формально. Пусть $G = (V, E)$ ориентированный граф с набором вершин V и набором рёбер $E \subseteq V \times V$. Для каждой вершины определим множество $In(V_i)$ как набор вершин V_j таких, что из них существует ребро, направленное к V_i. За $Out(V_i)$ обозначим множество вершин, в которые существуют ребра из V_i. Вес вершин будем определять по следующей формуле:

$$S(V_i) = (1 - d) + d \cdot \sum_{V_j \in In(V_i)} S(V_j)$$

где $d \in [0; 1]$ - понижающий параметр. Работа начинается с произвольных значений $S(V_i)$ для каждой вершины. Вычисление продолжается до тех пор, пока разница значений между итерациями не станет меньше заранее определенного порога.

Чтобы применить такой подход к тексту нужно построить граф, который будет этот текст представлять. Необходимо определить связи между элементами и другими элементами текста, и сами элементы, с которыми будет работать алгоритм. Ими могут быть отдельные слова, фразы или целые предложения. Можно определить следующие основные этапы для построения представления:

1. Определить единицы текста, с которыми будет работать алгоритм, и добавить их в граф в качестве вершин
2. Определить связи между этими элементами. Они будут представлены в графе ребрами. Ребра могут быть ориентированными или нет, это определяется задачей
3. Провести достаточное количество итераций для того, чтобы алгоритм "сошелся"
4. Отсортировать вершины по их весу. Этот вес используется для дальнейшей работы с элементами

Такое определение позволяет использовать подход для широкого круга задач. Далее рассмотрим приложение этого алгоритма для автоматического извлечения ключевых слов.

На начальном этапе необходимо определить множество основных элементов. В нашей задаче этим множеством является подмножество множества всех слов текста. Дело в том, что не все слова несут полезную информацию. Например, шумовые слова обычно удаляются из текста перед анализом. Авторы алгоритма предлагают отбирать слова в зависимости от части речи. По результатом экспериментов они установили, что наилучших показателей удалось добиться, если отбирались только существительные и прилагательные. Слова, которые остались после лингвистической фильтрации, добавлялись в граф в качестве вершин. Ребро между вершинами добавляется в том случае, если слова, соответствующие им, встречаются в тексте на расстоянии не более N слов, в окне вхождения. Экспериментально было выбрано значение $N = 2$. В результате получается не взвешенный неориентированный граф. На начальном этапе каждой вершине присваивается единичный вес. Далее возникает задача поиска наиболее значимых вершин в графе, для этого используется следующий алгоритм описанный выше, с тем замечанием, что $In(V_i) = Out(V_i)$, поскольку в данному случае граф неориентированный. После того, как для каждой вершины определен вес начинается этап отбора ключевых фраз. Выбираются первые T с наибольшими весами. Значение T может быть как фиксированным числом, так и зависеть от параметров, например размера текста. Слова, которые в исходном графе были представлены соседними вершинами объединяются в последовательности.

Алгоритм тестировался на наборе абстрактов к статьям. Эталонный набор ключевых слов для них был определен заранее вручную. Оценки эффективности алгоритма по результатам экспериментов:

Precision	32.1%
Recall	36.2%

Стоит отметь тот факт, что в силу определения алгоритма, количество вершин графа зависит от количество различных слов. Для больших текстов выполнения алгоритма может занять много времени.

3.2 PageRank on Synonym Networks[7]

Частично решением проблемы роста графа является объединение слов в классы эквивалентности. Кроме ограничение роста, такое решение может повысить эффективность работы алгоритма. Одним из способов определения классов эквивалентности является семантический способ. При наличии словаря синонимов, слова можно объединить в классы эквивалентности по их смысловому значению. Каждый такой класс можно представить уникальным номером. Каждое слово в тексте заменяется на такой уникальный номер, который соответствует группе эквивалентности данного слова. Применяя алгоритм, описанный в предыдущем пункте к измененному тексту, авторы алгоритма смогли добиться следующих результатов:

Precision	40.9%
Recall	73.9%

Авторы тестировали своё решение на наборе блогов с обозначенными ключевыми словами. Авторы отбирали те блоги, в которых основная часть информации представлялась в виде текста, а количество обозначенных тегов было не меньше трех. Далее тексты были разбиты на группы в зависимости от темы. Результатом работы алгоритма являются не сами слова, а набор идентификаторов классов эквивалентности, поэтому для получения оценок эталонный набор ключевых слов представлялся в таком же виде.

3.3 RAKE[2]

RAKE (Rapid Automatic Keyword Extraction) – сравнительно эффективный алгоритм, который применяется для одиночных документов. Он может быть использован для анализа документов из различных предметных областей. Кроме того, этот алгоритм никак не зависит от структуры документа. Данный алгоритм основывается на том, что в ключевых словах редко встречаются шумовые слова и знаки пунктуации. Нужно отметить, что такие слова как правило считаются не информативными и не участвуют в индексации в системах информационного поиска. Для работы алгоритма потребуется список таких слов

Кроме списка стоп-слов алгоритм на вход получает два списка разделителей: для разделения текста на фразы и для разделения на отдельные слова. Работа алгоритма начинается с разбиения текста на фразы – кандидаты в ключевые слова. Чтобы получить этот список используются шумовые слова и список разделителей для фраз. В результате получается набор последовательностей, содержащих только значимые слова.

После того, как был получен набор кандидатов, необходимо вычислить вес для каждого из них. Для этого каждая фраза разбивается на отдельные слова с помощью набора разделителей, который был подан на вход. В результате получается набор слов. Для каждого слова вычисляется вес. Авторы предлагают использовать частоту и степень для вычисления веса слова. Степень слова ($deg(w)$) определяется как суммарное количество слов, из которых состоят фразы, в которых оно содержится. Частота ($freq(w)$) определяется как количество фраз, в которые входит рассматриваемое слово. Вес слова определим как отношение степени слова к его частоте: $s(w) = \frac{deg(w)}{freq(w)}$. Чтобы вычислить вес фразы нужно сложить веса слов, из которых она состоит. В качестве результата предлагается отбирать T кандидатов с наибольшим весом.

Авторы исследовали эффективность алгоритма на наборе абстрактов к статьям. Эталонные ключевые слова были обозначены вручную для каждого абстракта. В экспериментах авторы использовали значение $T = \frac{1}{3}$, т.е. в качестве ключевых слов отбиралась треть кандидатов. Следует

отметить, что эффективность алгоритма во многом зависит от набора стоп-слов, который используется для получения кандидатов.

stoplist	Precision	Recall
KA stoplist (df $>$ 10)	33.7%	41.5%
Fox stoplist	26.0%	42.2%

В таблице представлены результаты экспериментов для двух наборов стоп-слов. Можно заметить, как сильно отличается точность для каждого из них.

3.4 Ранжирование в зависимости от позиции[6]

Алгоритм основан на том наблюдении, что позиция в тексте играет ключевую роль в значимости слова. Так, например, предполагается, что слово, которое встречается во вступительном и завершающем параграфах, будет нести больший смысл, чем то, которое встречается в промежуточном. Аналогичные рассуждения применяются и для предложений в рамках одного параграфа. Заголовок текста считается параграфом.

Авторы предлагают новый метод - "Позиционное взвешивание"(Position Weight). Он заключается в следующем. Пусть $pw(t_i)$ вес терма t в позиции i, определим его как

$$pw(t_i) = pw(t_i, p_j) * pw(t_i, s_k) * pw(t_i, w_r)$$

, где первый множитель отвечает за вес слова в рамка параграфа, второй - в рамках предложения, третий - в рамках формы слова. Тогда вес терма t в документе d будет вычисляться как сумма весов по всем вхождениям терма:

$$PW(t, d) = \sum_{i=1}^{m} pw(t_i)$$

Вес $pw(t_i, p_j)$, слова в рамках параграфа, определяется следующим образом:

1. Если параграф - заголовок, то $pw(t_i, p_j) = 4Unit_{par}$
2. Если параграф - подзаголовок, то $pw(t_i, p_j) = 3.5Unit_{par}$
3. Если параграф является заключительным или вступительным, то $pw(t_i, p_j) = 3Unit_{par}$
4. Если параграф является переходным, то $pw(t_i, p_j) = 2Unit_{par}$
5. В остальных случаях $pw(t_i, p_j) = 1Unit_{par}$

Вес $pw(t_i, s_j)$, слова в рамках предложения, определяется следующим образом: 0.5

1. Если предложение - заголовок, то $pw(t_i, s_k) = 4Unit_{sen}$
2. Если предложение является заключительным или вступительным, то $pw(t_i, s_k) = 3Unit_{sen}$
3. Если предшествует заключительному, следует сразу, после вступительного или является переходным, то $pw(t_i, s_k) = 2Unit_{sen}$
4. Если слово описывает примеры, то $pw(t_i, s_k) = 0Unit_{sen}$
5. В остальных случаях $pw(t_i, s_k) = 1Unit_{sen}$

Вес $pw(t_i, w_d)$, слова в рамках формы, определяется следующим образом:

1. Если слово начинается с заглавной буквы и содержит цифры, то $pw(t_i, w_d) = 3Unit_{word}$
2. Если слово начинается с заглавной буквы, то $pw(t_i, w_d) = 2Unit_{word}$
3. В остальных случаях $pw(t_i, w_d) = 1Unit_{word}$

На вход алгоритм также принимает набор слов, которые позволяют определить тип предложения.(Например, заключительное предложение будет содержать фразы на подобии: "в итоге "в заключение"). Также в эксперименте использовались следующие значения: $Unit_{par} = Unit_{sen} = Unit_{word} = 1$

Из определения видно, что алгоритм сильно зависит от структуры документа. Заметим, что в случае нарушения этих предположений (нет разбиения на параграфы, а предложения не содержат специальных слов, позволяющих определить их тип), алгоритм сводится к подсчёту количества вхождений слова.

3.5 Статистический подход

3.5.1 Взвешивание одиночных слов

Кроме частоты слова из текста можно также получить информацию о распределении слова. Авторы подхода предлагают использовать эту информацию для определения значимости. Метрики, предложенные в статье[5], опираются на феномен притяжения ключевых слов текста. Так, например, авторы приводят распределение расстояний до ближайшего соседа для слова "NATURE" в книге "The Origin of Species" by Charles Darwin (Рис. 1). На графике можно увидеть, что распределение расстояний напоминает экспоненциальное. У шумовых слов напротив такое свойство проявляется гораздо слабее. Таким образом, ключевые слова склонны образовывать кластеры в тексте.

Авторы статьи[5] предлагают несколько метрик, позволяющих оценить значимость каждого слова в тексте.

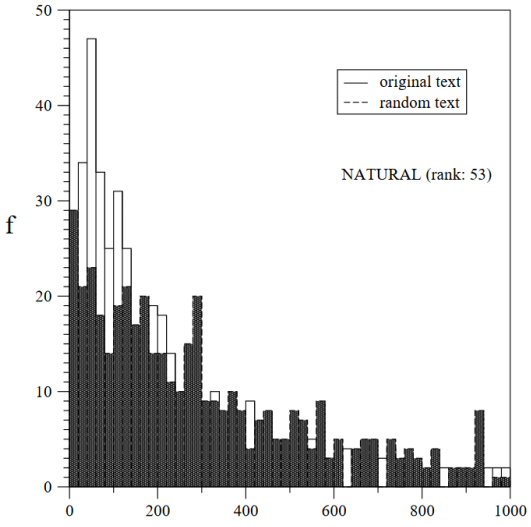


Рис. 1: Распределение расстояний до ближайшего соседа

σ-индекс

Пусть каждая позиция в тексте занумерована числами от 1 до N. Рассмотрим некоторое слово из текста. Пусть оно входит в текст $n + 2$ раз. Обозначим его последовательные вхождения за $\{t_0, t_1, \ldots, t_{n+1}\}$. Тогда среднее расстояние между вхождениями вычисляется по формуле:

$$\mu = \frac{1}{n+1} \sum_{i=0}^{n} (t_{i+1} - t_i)$$

. Тогда среднее отклонение расстояний между соседями будет вычисляться по формуле:

$$s = \sqrt{\frac{1}{n+1} \sum_{i=0}^{n} ((t_{i+1} - t_i) - \mu)^2}$$

. Для того, чтобы избавиться от влияния частоты вхождения слова на его вес, будем считать его равным $\sigma = \frac{s}{\mu}$. Действительно, если слово распределение слова в тексте неоднородно, то значение среднего отклонения s будет велико, т.е. для ключевых слов вес будет принимать большие

значения, чем для не ключевых. С другой стороны, вес сильно зависит от позиции слов – изменение позиции одного слова может привести к сильному изменению его веса. Кроме того, большие значения σ не всегда подразумевает наличие кластера: большой кластер может быть разбит на более мелкие без существенного изменения веса σ

Г-индекс

Тогда как вес σ учитывает расстояние между ближайшими соседями, Г-индекс использует среднее расстояние до соседей для каждого вхождения t_i:

$$d(t_i) = \frac{(t_{i+1} - t_i) - (t_i - t_{i-1})}{2} = \frac{t_{i+1} - t_{i-1}}{2}$$

. Будем называть вхождение t_i кластерной точкой, если $d(t_i) < \mu$. Тогда значимость слова будет зависеть от количества таких вхождений. Определим кластерную меру для каждой позиции t_i как

$$\gamma(t_i) = \begin{cases} \frac{\mu - d(t_i)}{\mu}, & if\ d(t_i) < \mu \\ 0, & else \end{cases}$$

Тогда вес слова будем вычислять по формуле:

$$\Gamma = \frac{1}{n} \sum_{i=1}^{n} \gamma(t_i)$$

. Такой вес более стабилен, чем σ, но требует больше времени для вычисления.

3.5.2 Извлечение ключевых фраз

Описанные выше способ взвешивания даёт возможность ранжирования только одиночных слов. В статье[4] предлагается несколько способов обобщения метрик на фразы из двух слов.

Способ 1

Ранжировать целые последовательности слов. Вес последовательности вычисляется аналогично одиночному слову. Проблема заключается в том, что частота нужных фраз будет недостаточной для анализа[4]

Способ 2

Составить все возможные уникальные фразы заданной длины. Тогда вес фразы будет вычисляться как сумма весов слов, которые его составляют. Проблема этого подхода в том, что количество фраз растёт экспоненциально с увеличением длины фразы.

3.6 Выводы

Рассмотренные алгоритмы демонстрируют различные подходы к извлечению ключевых слов. Каждый из них имеет свои преимущества и недостатки. Наибольший интерес вызывают те методы, которые меньше всего полагаются на какие-либо языковые особенности.

В рамках работы эффективность алгоритмов оценивается в терминах полноты и точности. Тем не менее, время работы алгоритма тоже играет не последнюю роль.

В работе будут детально рассмотрены два подхода RAKE и статистический подход. Статистический подход выделяется своей универсальностью и отсутствием необходимости в дополнительных данных и инструментах. Слабой стороной алгоритма является извлечение фраз. Второй

способ является наиболее эффективным, но в то же время он очень требователен к памяти. В работе будет показано, как решить эту проблему.

RAKE выделяется своей языковой независимостью и быстротой работы. В то же время он сильно зависит от набора шумовых слов. В работе будет показано, как решить эту проблему, и будет дана оценка на потребление памяти в таком решении.

4 Эксперименты

4.1 Описание данных

В качестве анализируемых данных была выбрана книга "Теорема Абеля в решениях и задачах"Алексеева В.Б. Текст содержит 39519 слов, из них 1576 уникальных. Алфавитный указатель был взят в качестве эталонного набора ключевых слов. Выбор анализируемых данных объясняется, во-первых, большим объёмом данных, что позволит дать более точные оценки эффективности алгоритмов, и, во-вторых, требованиями статистического подхода к количеству вхождений анализируемых слов (для небольших текстов, например абстрактов, такое условие может не выполниться ни для одного слова).

4.2 Получение оценки

Эффективность алгоритмов оценивается по метрикам *Precision* и *Recall*. Результатом работы каждого из алгоритмов является набор слов. Для получения оценки *Precision* необходимо определить, сколько ключевых фраз в результирующем наборе. Будем считать, что фраза является ключевой, если она содержит полностью какую-нибудь фразу из эталонного набора, при этом порядок слов неважен. Для получение оценки *Recall*, необходимо определить, сколько ключевых фраз из эталонного набора попали в результирующий. Также будем считать, что фраза содержится в наборе, если она целиком без учета порядка слов содержится в какой-то фразе из набора. Кроме того, все слова в наборах приведены в нормальную форму, что решает проблему разных форм слова.

4.3 RAKE

Для работы алгоритма необходим список стоп-слов. Был взят список, находившийся в открытом доступе с сайта ranks.nl. Сначала были получены оценки для оригинального алгоритма. На этом этапе, все кандидаты выбирались в качестве ключевых слов. Всего алгоритм извлек 5621 ключевую фразу. После приведения всех слов во фразах к нормальной форме, уникальных оказалось 4320. Оценки *Precision* и *Recall* оказались следующими:

Precision	30.4%
Recall	74.2%

Проблемы исходного алгоритма очевидны. Во-первых, около пятой части фраз, попавших в вывод, отличались только формой, а значит не являлись уникальными. Во-вторых, в вывод попали длинные фразы. Решение первой проблемы - использование стеммера на этапе подсчёта весов слов и на этапе формирования кандидатов в ключевые слова. В экспериментах использовался SnowballStemmer из библиотеки nltk версии 3.2.2 для языка Python. После применения стеммера были полученные такие оценки:

Precision	31.8%
Recall	74.2%

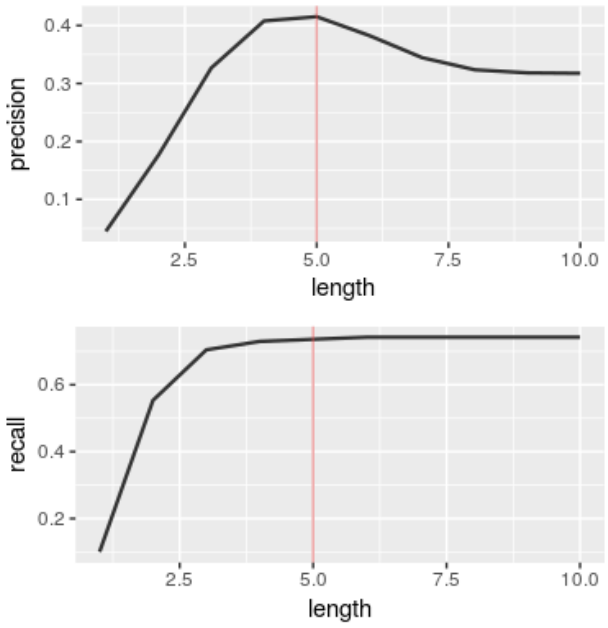


Рис. 2: Точность и полнота алгоритма в зависимости от ограничения на длину фразы

Вывод алгоритма содержал 4320 фраз, каждая являлась уникальной в рамках приведения к нормальной форме. Можно заметить, что использование стеммера дало небольшой прирост только к точности метода. Понятно, что использование стеммера не улучшило разбиение на фразы: в выводе всё ещё есть длинные последовательности слов. Длина наибольшей из таких последовательностей равна 10. Хотелось бы получить в выводе более короткие, но ёмкие фразы. Решением проблемы является разбиение длинных фраз на более короткие. Это было сделано следующим образом: из слов, из которых состоит длинная фраза, составляются все возможные короткие фразы заданной длины без учёта порядка слов. Также было наложено ограничение и на значение степени(degree) слова, чтобы избежать перекоса в ранжировании слов. Теперь значение степени не может превышать значения длины последовательности, которая попадает в вывод. Одинаковые фразы, которые могли быть получены при таком подходе, отбрасываются на этапе подсчёта весов фраз.

На Рис. 2 представлены результаты экспериментов по исследованию зависимости полноты и точности алгоритма от ограничения на длину фразы.

Наибольшее значение точности было достигнуто при максимальной длине фразы, равной 5. При этом полнота алгоритма для этого значения не максимальна, но близка к ней. Ниже приведена таблица для точности, полноты и количества слов при соответствующих ограничениях.

Max. length	Precision	Recall	NWords
4	40.8%	73.0%	9327
5	41.5%	73.6%	7293

Имеет смысл рассматривать только такие ограничения на длину, поскольку при этих значениях точность алгоритма достигает наибольших значений, при значениях полноты близких к максимальному. Можно заметить, что количество слов, извлекаемых алгоритмом, резко возросло, при этом, возросло и количество правильно извлеченных слов, поскольку наблюдается и прирост точности.

Такой способ генерации кандидатов в теории может вызвать экспоненциальный рост их количества. Под длиной фразы будем понимать количество слов, из которых она состоит. Пусть k – ограничение на максимальную длину фразы, m - максимальная длина фразы, которая попала в вывод алгоритма. В силу определения алгоритма, фразы, полученные после разбиения с помощью стоп-слов и списка разделителей, не имеют пересечений в тексте, т.е. для любых двух различных фраз, их позиции в тексте не будут пересекаться. Пусть N – количество слов в тексте. Тогда таких фраз не может быть больше, чем $\frac{N}{m}$. Будем считать, что $m > k$. При разбиении не учитывается порядок слов, поэтому количество новых фраз можно вычислить как количество сочетаний без повторений:

$$C_m^k = \frac{m!}{(m-k)! \cdot k!}$$

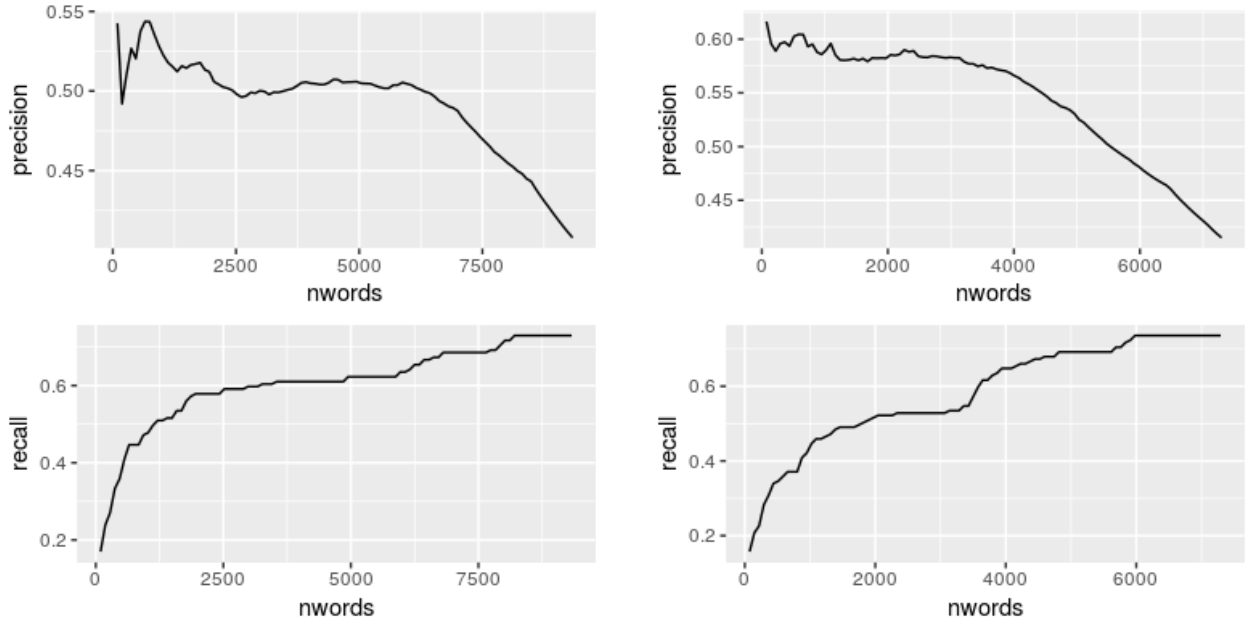
Таким образом можно дать следующую оценку нового количества фраз:

$$\frac{N}{m} \cdot C_m^k$$

Прирост пропорционален C_m^k. По определению алгоритма, длина ключевой фразы не может превосходить длины предложения. Случайное предложение состоит из 10-11 слов[10], а это значит, что среднее значение m не будет превосходить этого числа. Таким образом, можно сделать вывод, что рост числа кандидатов будет в разумных пределах для случайных текстов.

Ограничивая размер результирующего набора, можно повысить точность алгоритма. Но отбрасывая часть результатов может уменьшится полнота. На Рис.3 приведена зависимость точности и полноты от количества отобранных кандидатов.

Исходя из того, что точность алгоритма растёт с уменьшение доли отобранных фраз, можно сделать, что ключевые фразы действительно получили наибольший вес. Поэтому можно сказать, что алгоритм RAKE может использоваться для автоматического извлечения ключевых слов из текста.



(a) Ограничение на длину фразы = 4

(b) Ограничение на длину фразы = 5

Рис. 3: Зависимость полноты и точности от количества отобранных кандидатов

4.4 Статистический подход

Статистический подход даёт способ измерения значимости отдельных слов. Тем не менее, ключевые слова могут представлять собой и фразы. В связи с этим возникает вопрос, каким образом получить набор кандидатов. Решения предложенные в статье[4] показали хорошие результаты, но они требуют большое количество памяти для хранения всех кандидатов. Так, например, количество возможных фраз из двух слов (без учета порядка) будет равно C_N^2, где N - количество уникальных слов в тексте. Так, для используемого в работе текста количество уникальных слов ≈ 1500, а значит таких фраз будет порядка 10^6. Если анализировать только фразы из двух слов, то добиться максимальной полноты не удасться, поскольку максимальная длина ключевой фразы – 3 слова. Количество все возможных фраз из трёх слов будет уже порядка 10^9.

На самом деле, нет необходимости хранить всех кандидатов, поскольку в качестве результата работы алгоритма будут отобраны только первые T кандидатов. Так как вес фразы вычисляется как сумма весов слов,то имея набор взвешенных слов, достаточно отобрать первые T фраз с наибольшим весом.

Отбор кандидатов в ключевые слова

Сначала рассмотрим случай отбора фраз из двух слов. Эту задачу можно свети к следующей. Даны два отсортированных массивы $A[1 \dots n]$ и $B[1 \dots n]$. Мы хотим вывести все n^2 сумм вида $A[i] + B[j]$ в убывающем порядке. Существует решение задачи с массивами, которое требует $O(n^2 log(n))$ времени и $O(n)$ памяти.

Описание решения задачи о массивах

Заведём max-кучу, в которой будем хранить пару: сумму $A[i] + B[j]$ и пару индексов, которые эту сумму задают (i, j). Элементы кучи сортируются по записанным в них суммам.

Запишем в кучу пары: $j = 0 \dots n : (A[0] + B[j], (0, j))$ - суммы первого элемента массива A с каждым из элементов массива B с соответствующими индексами.

Шаг алгоритма заключается в том, что из кучи извлекается первый элемент – это элемент с наименьшей суммой в куче. Пусть эту сумму задавали индексы (i, j). Выводим сумму, а в кучу записываем новый элемент: $A[i + 1] + B[j]$ с соответствующей парой индексов: $(i + 1, j)$.

Когда элементы массивов закончились, то выводится оставшееся содержимое кучи в порядке возрастания сумм. Из определения решения справедливость оценок затрат времени и памяти очевидны.

Корректность решения

Для доказательства корректности удобно рассмотреть все возможны суммы в виде квадратной матрицы. Назовём эту матрицу C: $c[i, j] = A[i] + B[j]$. Поскольку элементы массивов A и B отсортированы, то можно заметить, что элементы матрицы, убывают слева-направо и сверху-вниз. Иными словами, $\forall i, j : c[i, j] \geq c[i + 1, j]$ и $c[i, j] \geq c[i, j + 1]$.

На каждом шаге поддерживается инвариант: в куче хранятся максимальные элементы столбцов, которые ещё не попали в вывод, по одному элементу из каждого столбца. Поскольку минимальный элемент, который должен попасть в вывод находится в одном из столбцов, это доказывает корректность.

Сведение к задаче о кандидатах

Пусть элементы массивов - это пары вида $(score, word)$, где $score$ - вес слова $word$. Массивы отсортированы в порядке уменьшения $score$. Если определить сумму двух пар как $(score_1, word_1) + (score_2, word_2) = (score_1 + score_2, word_1 * word_2)$, где $*$ - операция конкатенации, то выведение первых T пар в порядке убывания суммы $score_1 + score_2$ – есть исходная задача.

Если оставить решение без изменения, то в вывод попадут все возможные пары с учетом порядка, и их придётся отфильтровать. Требование $i + 1 < j$ при добавлении в кучу гарантирует, что в вывод попадут только уникальные фразы.

Обобщение для случая трёх слов

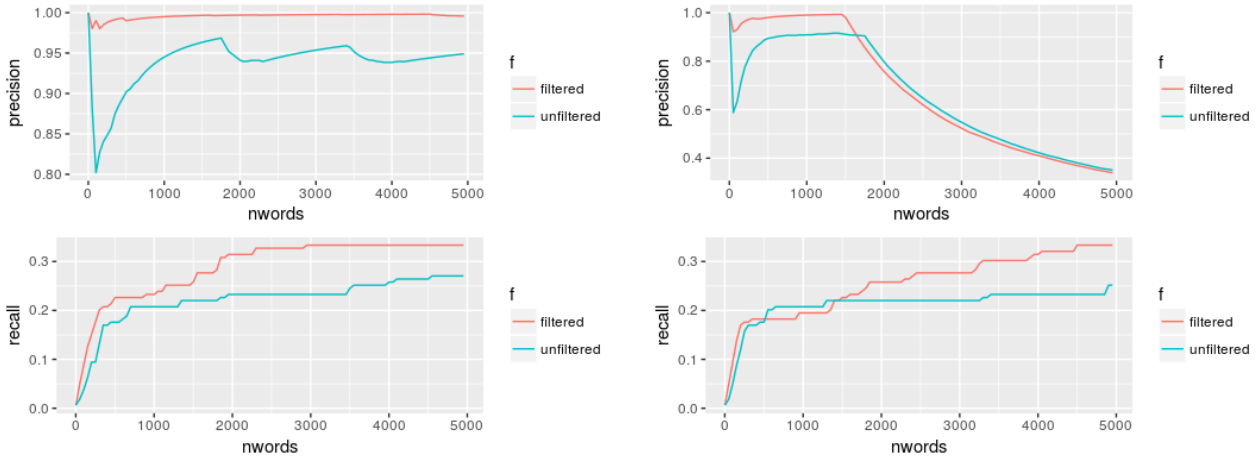
Теперь есть три массива $A[1 \dots n]$, $B[1 \dots n]$ и $C[1 \dots n]$. Задача сводится к задаче с двумя массивами, если заменить массивы B и C на массив $B'[1 \dots n^2]$, элементами которого являются суммы

всех возможных пар элементов массивов B и C. Задача аналогично сводится к задаче о кандидатах. Оценка затрат времени будет $O(n^3 \cdot log(n))$, а памяти – $O(n^2)$. Аналогично можно сделать обобщение для произвольного количества массивов.

Таким образом, удалось сократить затраты памяти на порядок. Для случая двух слов: $O(n^2) \longrightarrow O(n)$, для случая трёх слов: $O(n^3) \longrightarrow O(n^2)$.

Оценка эффективности алгоритма

Перед тем как вычислять вес, каждое слово было приведено в нормальную форму с помощью SnowballStemmer библиотеки nltk версии 3.2.2. Таким образом, слова, отличавшиеся только формой, были объединены в классы эквивалентности. Далее веса вычислялись для таких классов эквивалентности. В качестве меры значимости был выбран Г-индекс. Кроме того, на этапе отбора слов для взвешивания было решено использовать фильтр по длине слова: те слова, длина которых оказывалась меньше заданного порогового значения не участвовали во взвешивании и в построении кандидатов в ключевые слова. В экспериментах с фильтрацией пороговое значение для длины слова было равно 3. Кроме того было проведено сравнение эффективности алгоритмов в зависимости от длины кандидатов. На Рис.4 приведены результаты экспериментов.



(a) Длина отобранных фраз равна 3. Синий график соответствует результатам алгоритма без использования фильтра по длине слов, красный - с фильтром

(b) Длина отобранных фраз равна 2. Синий график соответствует результатам алгоритма без использования фильтра по длине слова, красный - с фильтром

Рис. 4: Графики зависимости точности и полноты от количества отобранных кандидатов

На графиках видно, что использование фильтра дало заметный прирост к полноте и точности алгоритма, возвращающего фразы длины 3. Для алгоритма, который возвращает фразы длины 2, выигрыш зависит от количества отобранных фраз.

Рассматриваемые алгоритмы имеют очень высокую точность. Это объясняется построением кандидатов в ключевые фразы. Используемый вес очень точно ранжирует отдельные слова, и слова с наибольшим весом с высокой вероятностью оказываются ключевыми, если эталонный набор содержит одиночные ключевые слова, или содержаться в ключевых фразах эталонного набора. В эталонном наборе есть ключевые фразы состоящие из одного слова. Поэтому первые кандидаты будут являться ключевыми с высокой вероятностью, поскольку будут составляться из таких слов. На графике для фраз длины 2 можно заметить резкий спад точности с некоторого момента. Это объясняется тем, что начинают создаваться фразы из не ключевых слов. Такой же провал есть и у фраз длины 3, но он наступает гораздо позже и менее заметен в силу большого

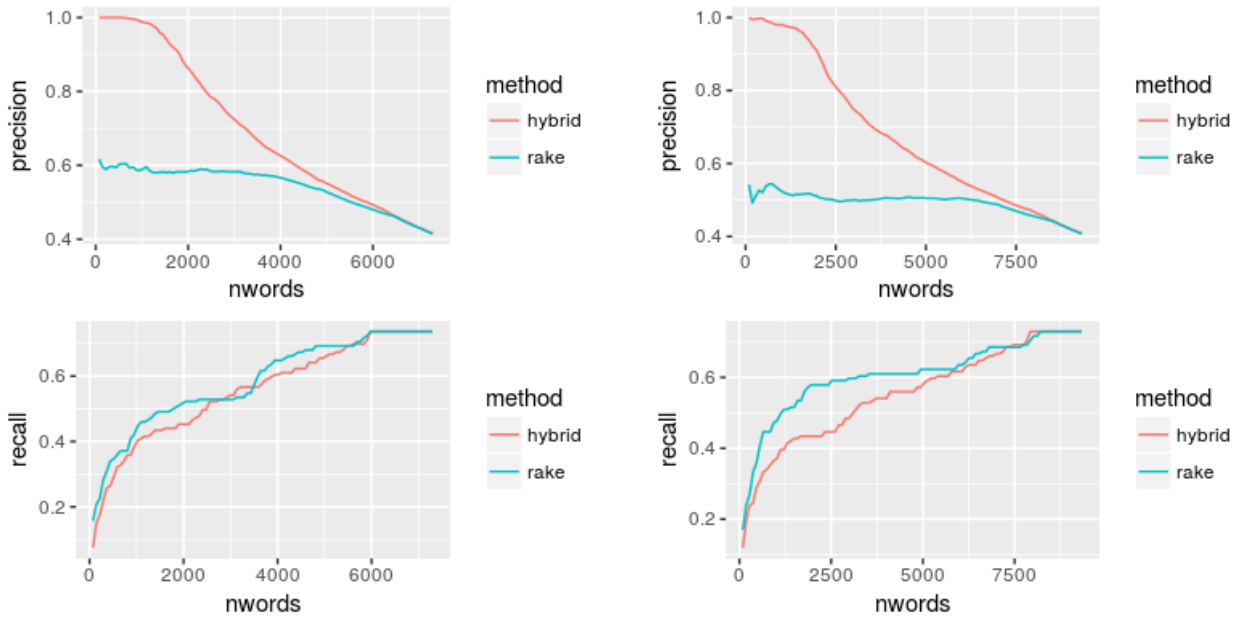
количества слов.

По результатам экспериментов можно сделать вывод о том, что используемая метрика Г-индекс правильно ранжирует отдельные слова. Тем не менее, искусственно составленные фразы не могу обеспечить высокую полноту алгоритма.

4.5 Гибридный подход

Преимущество статистического подхода заключается в высокой точности ранжирования одиночных слов. А основным недостатком является сложность составления фраз. Алгоритм RAKE напротив, позволяет составить фразы, которые обеспечивают сравнительно высокую полноту, но их ранжирование не позволяет добиться высокой точности при ограничении на количество кандидатов. В связи с этим возникает идея применения метрики Г-индекс для ранжирования одиночных слов, но вес с помощью них определять для фраз, которые получаются в результате работы алгоритма RAKE. На Рис. 5 представлено сравнение соответствующих методов.

На графиках видно, что точность метода RAKE с использованием метрики Г-индекс гораздо выше, чем с метрикой $\frac{deg}{freq}$. Можно видеть, что для обоих ограничений на длину фразы при уменьшении количества отобранных слов точность стремится к 1. Тем не менее, использование новой метрики вызвало снижение полноты в обоих случаях, особенно для случая с ограничением на длину фразы равным 4.



(a) Ограничение на длину фразы = 5

(b) Ограничение на длину фразы = 4

Рис. 5: Графики изменения точности и полноты для методов RAKE и Hybrid в зависимости от количества отобранных слов

5 Заключение

В работе было показано, что при определенных дополнениях эффективность применения алгоритма RAKE и статистического подхода к русскоязычным текстам не уступает их эффективности для англоязычных. Кроме того, в работе была предложена идея гибридного метода, который использует метрику Г-индекс для взвешивания фраз, которые были получены в соответствии с

определением алгоритма RAKE. И было показано, что такой алгоритм превосходит по точности RAKE при уменьшении числа отобранных фраз.

6 Список литературы

1. Анализ данных и процессов[Текст] / А.А. Барсегян, М.С. Куприянов, И.И. Холод, и др. – БХВ-Петербург, 2009.-510 с.

2. Michael Berry. Text Mining: Applications and Theory[Текст] / Michael Berry, Jacob Kogan – 2010, John Wiley and Sons, Ltd.-205 с.

3. Rada Mihalcea, Paul Tarau. TextRank: Bringing Order into Texts // In Proceedings of EMNLP 2004 (ed. Lin D and Wu D), pp. 404–411

4. Siddiqi, S., Sharan, A. Keyword and keyphrase extraction from single Hindi document using statistical approach // 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN) 2015, pp. 713-718

5. Juan P. Herrera, Pedro A. Pury. Statistical Keyword Detection in Literary Corpora // The European Physical Journal B, 2008, pp. 135-146.

6. Xinghua Hu, Bin Wu. Automatic Keyword Exctraction Using Linguistic Features //Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06), 2006, pp. 19-23

7. Zhengyang Liu, Jianyi Liu, Wenbin Yao, Cong Wang. Keyword Extraction Using PageRank on Synonym Networks // 2010 International Conference on E-Product E-Service and E-Entertainment, 2010, pp. 1-4

8. Jones S., Paynter G. Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications // Journal of the American Society for Information Science and Technology, 2002

9. Gutwin C, Paynter G, Witten I, Nevill-Manning C., Frank E. Improving browsing in digital libraries with keyphrase indexes // Decision Support Systems 27(1–2), 1999, pp. 81–104

10. С.А. Шаров. Частотный словарь [Электронный ресурс] - URL: http://www.artint.ru/projects/frqlist.ph (дата обращения 20.05.2017)

Michael BerryMichael BerryMichael BerryMichael BerryMichael BerryMichael BerryMichael BerryMichael Berry